Amendments to the Claims

1. (Previously Amended) A method comprising:

calling a scheduling driver to start an Input/Output (I/O) request to a device for an application, the device being one of a plurality of different types of devices useable by an application;

determining if the device is busy; and

if the device is not busy,

providing an estimated processing time (EPT) for the I/O request to be completed for the application, wherein the application sleeps for the estimated processing time.

2. (Original) The method of claim 1, wherein determining if the device is busy comprises determining whether a locked flag is set, if the locked flag is set the device is busy and if the locked flag is not set the device is not busy.

3. (Original) The method of claim 1, further comprising, setting a locked flag if the device is not busy.

4. (Cancelled)

5. (Previously Amended) The method of claim 1, further comprising, calling the scheduling driver to obtain I/O operation results after sleeping for the estimated processing time and determining if the I/O request has been completed.

6. (Original) The method of claim 5, further comprising, clearing a locked flag if the I/O request has been completed.

7. (Previously Amended) The method of claim 5, further comprising, providing the I/O operation results from the I/O request if the I/O request has been completed.

8. (Original) The method of claim 5, further comprising, sleeping for a timer tick interval if the I/O request has been completed.

9. (Previously Amended) The method of claim 5, further comprising, calculating an estimated processing time remaining (EPTR) for the I/O request to be completed, if the I/O request has not been completed, and providing the estimated processing time remaining (EPTR).

10. (Previously Amended) The method of claim 9, further comprising:

sleeping for the estimated processing time remaining (EPTR);

calling the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

determining if the I/O request has been completed.

11. (Previously Amended) The method of claim 10, further comprising:

determining if the I/O request has been completed and calculating an estimated processing time remaining (EPTR) for the I/O request to be completed, if the I/O request has not been completed;

sleeping for the estimated processing time remaining (EPTR);

calling the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

if the I/O request has not been completed,

repetitively performing the above operations until the I/O request has been completed.

12. (Previously Amended) The method of claim 1, further comprising calculating an estimated amount of time left (EATL) until the device will be available if the device is busy, and providing the estimated amount of time left (EATL).

13. (Previously Amended) The method of claim 12, further comprising:

sleeping for the estimated amount of time left (EATL);

calling the scheduling driver to start the I/O request to the device after sleeping for the estimated amount of time left (EATL); and

determining if the device is still busy.

14. (Previously Amended) The method of claim 13, further comprising:

determining if the device is still busy and calculating the estimated amount of time left (EATL) until the device will be available, if the device is still busy;

sleeping for the estimated amount of time left (EATL);

calling the scheduling driver to start the I/O request to the device for the application, after sleeping for the estimated amount of time left (EATL); and

if the I/O request has not been started,

repetitively performing the above operations until the I/O request has been started.

15. (Previously Amended) A machine-readable medium having stored thereon instructions, which when executed by a machine, causes the machine to perform operations comprising:

calling a scheduling driver to start an Input/Output (I/O) request to a device for an application, the device being one of a plurality of different types useable by an application;

determining if the device is busy; and

if the device is not busy,

providing an estimated processing time (EPT) for the I/O request to be completed for the application, wherein the application sleeps for the estimated processing time.

16. (Original) The machine-readable medium of claim 15, wherein determining if the device is busy comprises determining whether a locked flag is set, if the locked flag is set the device is busy and if the locked flag is not set the device is not busy.

17. (Original) The machine-readable medium of claim 15, further comprising the operation of setting a locked flag if the device is not busy.

18. (Canceled)

19. (Previously Amended) The machine-readable medium of claim 15, further comprising the operations of calling the scheduling driver to obtain I/O operation results after sleeping for the estimated processing time and determining if the I/O request has been completed.

20. (Original) The machine-readable medium of claim 19, further comprising the operation of clearing a locked flag if the I/O request has been completed.

21. (Previously Amended) The machine-readable medium of claim 19, further comprising the operation of providing the I/O operation results from the I/O request if the I/O request has been completed.

22. (Original) The machine-readable medium of claim 19, further comprising the operation of sleeping for a timer tick interval if the I/O request has been completed.

23. (Previously Amended) The machine-readable medium of claim 19, further comprising the operations of calculating an estimated processing time remaining (EPTR) for the I/O request to be completed, if the I/O request has not been completed, and providing the estimated processing time remaining (EPTR).

24. (Previously Amended) The machine-readable medium of claim 19, further comprising the operations of:

sleeping for the estimated processing time remaining (EPTR);

calling the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

determining if the I/O request has been completed.

25. (Previously Amended) The machine-readable medium of claim 24, further comprising performing the operations of:

determining if the I/O request has been completed and calculating an estimated processing time remaining (EPTR) for the I/O request to be completed, if the I/O request has not been completed;

sleeping for the estimated processing time remaining (EPTR);

calling the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

if the I/O request has not been completed,

repetitively performing the above operations until the I/O request has been completed.

26. (Previously Amended) The machine-readable medium of claim 15, further comprising the operations of calculating an estimated amount of time left (EATL) until the device will be available if the device is busy, and providing the estimated amount of time left (EATL).

27. (Previously Amended) The machine-readable medium of claim 26, further comprising the operations of:

sleeping for the estimated amount of time left (EATL);

calling the scheduling driver to start the I/O request to the device after sleeping for the estimated amount of time left (EATL); and

determining if the device is still busy.

28. (Previously Amended) The machine-readable medium of claim 27, further comprising performing the operations of:

determining if the device is still busy and calculating the estimated amount of time left (EATL) until the device will be available, if the device is still busy;

sleeping for the estimated amount of time left (EATL);

calling the scheduling driver to start the I/O request to the device, after sleeping for the estimated amount of time left (EATL); and

if the I/O request has not been started,

repetitively performing the above operations until the I/O request has been started.

29. (Previously Amended) An apparatus comprising:

a processor having a memory connected thereto, the memory storing an application, a scheduling driver, the application calling the scheduling driver to start an Input/Output (I/O) request to a device, the device being one of a plurality of different types of devices useable by an application;

the scheduling driver,

determining if a device is busy; and

if the device is not busy,

providing an estimated processing time (EPT) for the I/O request to be completed for the application, wherein the application sleeps for the estimated processing time.

30. (Original) The apparatus of claim 29, wherein determining if the device is busy comprises determining whether a locked flag is set, if the locked flag is set the device is busy and if the locked flag is not set the device is not busy.

31. (Original) The apparatus of claim 29, wherein the scheduling driver sets a locked flag if the device is not busy.

32. (Canceled)

33. (Previously Amended) The apparatus of claim 29, wherein the application calls the scheduling driver to obtain I/O operation results after sleeping for the estimated processing time and determines if the I/O request has been completed.

34. (Original) The apparatus of claim 33, wherein the scheduling driver clears a locked flag if the I/O request has been completed.

35. (Original) The apparatus of claim 32 wherein the scheduling driver provides the I/O operation results from the I/O request to the application if the I/O request has been completed.

36. (Original) The apparatus of claim 32 wherein the application sleeps for a timer tick interval if the I/O request has been completed.

37. (Original) The apparatus of claim 32 wherein the scheduling driver calculates an estimated processing time remaining (EPTR) for the I/O request to be completed, if the I/O request has not been completed, and provides the estimated processing time remaining (EPTR) to the application.

38. (Previously Amended) The apparatus of claim 37, wherein the application:

sleeps for the estimated processing time remaining (EPTR);

calls the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

determines if the I/O request has been completed.

39. (Previously Amended) The apparatus of claim 38, wherein the application:

determines if the I/O request has been completed;

sleeps for the estimated processing time remaining (EPTR) calculated by the scheduling driver;

calls the scheduling driver to obtain the I/O operation results after sleeping for the estimated processing time remaining (EPTR); and

if the I/O request has not been completed,

repetitively performing the above operations until the I/O request has been completed.

40. (Original) The apparatus of claim 29, wherein the scheduling driver calculates an estimated amount of time left (EATL) until the device will be available to the application if the device is busy, and provides the estimated amount of time left (EATL) to the application.

41. (Original) The apparatus of claim 40, wherein the application:

sleeps for the estimated amount of time left (EATL);

calls the scheduling driver to start the I/O request to the device for the application after sleeping for the estimated amount of time left (EATL); and

determines if the device is still busy.

42. (Previously Amended) The apparatus of claim 41, wherein the application:

determines if the device is still busy;

sleeps for the estimated amount of time left (EATL) calculated by the scheduling driver;

calls the scheduling driver to start the I/O request to the device for the application, after sleeping for the estimated amount of time left (EATL); and

if the I/O request has not been started,

repetitively performing the above operations until the I/O request has been started.

43. (Newly Added) The method as recited in claim 1, wherein the scheduling driver implements a protocol using time estimates enabling the scheduling driver to be usable with a device that does not generate interrupts.

44. (Newly Added) The method as recited in claim 8, further comprising synchronizing a system clock with a clock associated with the scheduling driver, wherein the timer tick indicates an instant where the system clock and scheduling driver clock simultaneously generate an interrupt.

45. (Newly Added) The method as recited in claim 1, further comprising:

loading the scheduling driver into an operating system such that applications are capable of generating I/O requests to the device.

46. (Newly Added) The method as recited in claim 45, wherein the scheduling driver is a passive software component.

47. (Newly Added) The method as recited in claim 45, wherein a single instance of the scheduling driver is shared among a plurality of applications that access the device.

48. (Newly Added) The method as recited in claim 1, where a plurality of applications simultaneously generate device I/O requests.

49. (Newly Added) The method as recited in claim 8, further comprising:

specifying a zero time interval, by the driver;

sleeping for a timer tick interval, thereby yielding a time slice by the application; and

switching, by an operating system scheduler, the CPU to a next application, while allowing the application to remain runnable.

50. (Newly Added) The method as recited in claim 1, wherein the scheduling driver does not poll, thereby allowing critical execution sections to be exited quickly.

51. (Newly Added) The system as recited in claim 29, wherein the scheduling driver implements a protocol using time estimates enabling the scheduling driver to be usable with a device that does not generate interrupts.

52. (Newly Added) The system as recited in claim 36, wherein a system clock is synchronized with a clock associated with the scheduling driver, wherein the timer tick indicates an instant where the system clock and scheduling driver clock simultaneously generate an interrupt.

53. (Newly Added) The system as recited in claim 29, wherein the scheduling driver is loaded into an operating system such that applications are capable of generating I/O requests to the device.

54. (Newly Added) The system as recited in claim 53, wherein the scheduling driver is a passive software component.

55. (Newly Added) The system as recited in claim 53, wherein a single instance of the scheduling driver is shared among a plurality of applications that access the device.

56. (Newly Added) The system as recited in claim 29, where a plurality of applications simultaneously generate device I/O requests.

57. (Newly Added) The system as recited in claim 36, wherein if the scheduling driver specifies a zero time interval, the application sleeps for a timer tick interval, thereby yielding a time slice by the application, and an operating system scheduler switches to a next application, while allowing the application to remain runnable.

58. (Newly Added) The system as recited in claim 29, wherein the scheduling driver does not poll, thereby allowing critical execution sections to be exited quickly.